COMMUNICATION

CHAT SYSTEMS

The Chat Systems document encompasses all forms of chat communication and related systems within the game, including:

- Spatial Speech
- Chat Channels
- Instant Messaging
- Text Generation Options
- Says
- Moods
- Emotes
- Filtering
- Ignore List
- Special Features

Other forms of in-game communication such as Mail are covered in their respective documents. All configuration options should be stored on a per-character basis, and allow the player to copy the option sets between characters.

Vision

The goal of the Chat Systems is to provide myriad ways for players to communicate within the game, and game context for doing so.

- ❑ There must be a wide range of in-game communication systems.
- ❑ Players need tools to shape their own experiences.
- ❑ We must encourage and support a community.
- ❑ Amy Jo Kim's principles for community building:
  - o We will build flexible, extensible gathering places.
  - o We will design for a range of roles.
  - o We will encourage appropriate etiquette.
- ❑ Humor and personality will be important in the game design.
- ❑ We must encourage a sense of player ownership in the world.
- ❑ We must have full support for localization, including Unicode support on client and server.
- ❑ The database size should be minimized.
- ❑ Bandwidth costs must be minimized.
- ❑ Players should feel empowered to take action against bad apples in the player base. They should never feel just like victims.

Terms

Chat

A Chat is an indirect form of communication between two or more individuals, usually taking place in the context of a *Channel*. All listeners listening to a chat channel receive messages sent within that channel.

Tell

A Tell is a direct form of communication from one individual and targeting a character or group name. Tells can be quickly replied to from any channel with the /reply command. Tell output is treated like a channel, and a chat pane and pane selection tab for Tells can be displayed on the chat interface.

Say

Say commands communicate text through the Avatar using spatial speech, regardless of the active channel. Spatial speech output is treated like a channel, and a chat pane and pane selection tab for Says can be displayed on the chat interface.

System Details

It is expected that all chat systems will be localized in a manner established by the Lead Engineer.

Spatial Speech

Spatial speech 'emanates' from the location of the player's ground based Avatar (though skills may affect the location and volume), and is displayed in the primary avatar chat window.

Text may be displayed above the heads of avatars in the 3D view. If this proves technically or aesthetically impossible, or if the player disables the option, we can display a temporary, colored indicator showing that a player has spoken. The color of the indicator should match the color of text displayed in the spatial chat window.

Whispering in Spatial Speech

During spatial speech, whisper is a way to converse with other players adjacent to the whisperer. Players outside of hearing range may hear text indicating that whispering is going on, and may see the avatar's emote accordingly. Finally, avatar animations (emotes) may also be used to enhance the visibility of spatial whispering. A whisper example follows, below.

Whisper Example:

Reece types:
`/whisper You take the little guy!`

Reece reads:
`You whisper, 'You take the little guy!'`

Nick (adjacent to Reece) hears:
`Reece whispers, "You take the little guy!"`

Rich and Raph (guarding the door) read:
`You see Reece whispering.`

Note - The targeted version of whisper may read:
`You see Reece whispering to Nick.`

All may see:
`Reece's avatar playing the whisper animation.`

Chat Channels

All players joined or listening to a chat channel will see text sent in or to a Chat Channel. The commands associated with channels are similar to the IRC counterpart.

A player can join many chat channels. Only one joined chat channel can be viewed at a time. The currently viewed channel is called the Active Channel, and all other channels the player has joined are called the Inactive Channels. Functions supported by the channel system must include:

- Creating / Naming Channels
- Listing Joining / Leaving / Channels
- Setting Channel Topic
- Inviting / Kicking / Banning Users to Channels
- Moderating Channels, 'Op' modes, and Voice (the floor)
- Listing Channel Users

These channel commands should work within the active channel as well as accepting a channel ID option. (Example: /invite *name* [*channel*])

Sending a message in the active channel is as simple as entering text at the chat prompt. Sending a message to an inactive channel can be accomplished by entering the name of the channel as a command, followed by the text to send.

It is important to note that while the game will provide a number of predefined channels, players can create new channels, with a variety of names and topics, both moderated and unmoderated. The following examples include a predefined channel, #auction, and a player defined channel, #bounty. To avoid spamming, players cannot send messages to game-run channels that they have not joined. Players cannot create channels that override the predefined channels or any keywords.

Example for #auction Channel

```
/auction Lightsaber initialed 'L.S.' for sale! 50000 credits
```

Example for #bounty Channel

```
/bounty Dead OR Alive: Rebo Mok of Tatooine - 10000 creds!
```

Whispering in Channels

In addition to spatial whispering, we want to provide a way to whisper directly to individuals in the context of a chat channel or even in the world. This aspect of targeted whispering may require an additional command or command options. In any event, the goal is to provide targeted and spatial whispering. Ideally, we can also provide targeted whispering in a spatial setting that would be visible to other players via animating gestures on the avatars.

Instant Messaging

Instant messaging sends a message to an individual or group. Instant messages can be initiated from any chat command line, or through both a command-line or menu interface.

Instant messages appear either in the active channel, or possibly in a window. Alternatively, we may provide a way to activate windowed messages through another command, such as /chat or /msg. Similar to mIRC's /msg, this should open a chat window between the sender and receiver. Ideally, other players could be invited into this 'hidden' channel without having to create a public, topical channel. Like IRC clients, it would be nice to support multiple targets by separating target names with a special delimiter (comma, pipe, et. al.)

The /tell command, for example, should send the specified message to the targeted individual or group.

In response, receivers of the /tell can immediately /reply with their own message.

Some features for Chat and Instant Messaging include AFK mode and Answering Machine mode.

AFK

Players who leave their keyboard for extended periods may let others in the channel know that they are not available for the moment. The AFK message should be editable by the player, and possibly stored on the server for speed.

"Answering Machine" Service

Players should be able to enable an answering machine like service, complete with outgoing message, which would offer 'callers' the option of leaving a message. For instant messages, the default incoming message would be that original text sent. Storage of messages should be limited on the server side. Optional features could include dropping non-priority, non-friend, and non-reply messages when the incoming message queue is full.

Text Generation Options

At a chat prompt, a player may simply type text and it will be relayed to the other players in chat. The prose produced by entering text without a command preceding it implies neither mood nor method.

The three modes of text generation are Raw, Chat, and Prose. Assuming Luke's mood is set to happy (see the Mood section, below):

Luke types:
`Hello`

Those with Raw mode enabled read:
`Luke: Hello`

Those with Chat mode enabled read:
`"Hello," Luke says.`

Those with Prose mode enabled may read:
`"Hello," Luke says happily.`

In Prose Mode (default), chat messages are received by the chat system, which then produces grammatical variations for the reader, including sentence termination, auto-capitalization, or splitting sentences where it "seems" appropriate to the algorithm used.

In Raw Mode, the reader sees only the sender's name followed by a colon delimiter, white space, and the unedited message.

In Chat Mode, players read a light prose style without moods or say alternates, and only two grammatical variations, either:

`Player says, "-----"` or `"----," Player says.`

Say Commands

By default, the "/say" command uses the current "sticky" settings for both Say Methods and Moods. The default Say Method is "says" and the default mood is neutral, meaning no mood is implied in the generated prose. Moods and Say Methods is described in the following sections.

An alternative to the say command is the single quote ' at the *beginning* of the chat line or chat line parameter (see Moods.) White space between the ' and subsequent text may be deleted by the prose generator.

Luke types:
```
/say Hello
```

Everyone reads:
```
"Hello," Luke says.
```

Alternatively, everyone reads:
```
Luke says, "Hello."
```

Alternatively, those with raw more enabled read:
```
Luke: Hello
```

Luke types (note the optional say command):
```
'Hello. My name is Luke
```

In prose mode, can produce many variations:
```
"Hello. My name is Luke," says Luke.
"Hello," says Luke. "My name is Luke."
Luke says, "Hello. My name is Luke."
```

In raw mode (option), saying something in chat will reproduce the exact text typed by the sender on the receiver's client.

Luke types:
```
/say Hello
```

Clients in raw mode see:
```
Luke: Hello
```

Say alternates

In addition to /say and the ' (single quote) versions of the say command, other forms of say should include:

say	add	admit	apologise	apologize
assert	babble	cackle	carol	chatter
complain	coo	declare	demand	describe
drawl	drone	enthuse	exclaim	explain
groan	growl	grumble	grunt	howl
indicate	moan	mumble	mutter	note
plead	prattle	protest	quote	ramble
request	retort	scream	smirk	snarl
sneer	state	stutter	threaten	utter
wail	whimper	whine		

Luke types:
```
/whine But I was going into Toshi Station to pick up some power converters...
```

Everyone reads:
"But I was going into Toshi Station to pick up some power converters," Luke whines.
Alternatively, everyone reads:
Luke whines, "But I was going into Toshi Station to pick up some power converters…"

Alternatively, those with raw more enabled read:
Luke: But I was going into Toshi Station to pick up some power converters…

In prose mode, can produce other variations:
"But I was going into Toshi Station to pick up some power converters," whines Luke.

"But I was going into Toshi Station to pick up some power converters," Luke whines.

Luke whines, "But I was going into Toshi Station to pick up some power converters…"

Note: Yelling may be primarily a spatial speech operation, however it might also be convenient to support yelling as a say alternate in other chat channels as well, providing context in prose mode and journals.

"Sticky" Say Commands

Like moods (below), some say alternates can be made 'sticky' with the /saymode command. Their character will walk around grumbling or cooing from session to session.

Moods (and Mood Commands)

Provide the ability to present moods within the context of the interactive chat system and journal.

The default mood is neutral and provides no prose context.

Mood Commands

Support dozens of moods through the use of multiple chat commands, each with one or more grammatical variations.

Example (assumes prose mode):
A human types:
/gruff say Don't insult us. You just watch yourself. We're wanted men. I have the death sentence in twelve systems.

Everyone reads:
"Don't insult us," a human says gruffly. "You just watch yourself. We're wanted men. I have the death sentence in twelve systems."

Luke types:
/contrite ' I'll be careful then.

Everyone reads:
`"I'll be careful then," Luke says contritely.`

A human types:
`/angry say You'll be dead!`

Everyone reads:
`Angrily, a human says, "You'll be dead!"`

"Sticky" moods

Some players may want their characters to walk around happily or gruffly in all gaming sessions, or their character may wish to present a bad mood.

To see your current mood, type:
`/mood`

To see a list of moods, type:
`/help moods`

To return to a neutral mood, type:
`/mood -`

To set the default mood for a character, simply type:
`/mood moodname`

Examples of some of the many supported moods and prose follow:

- *angry*
  'angrily'
  'furiously'
  'with a cold, angry glare'

- *bored*
  'bored'
  'uninterested'
  'bored stiff'

- *drunk*
  'drunkenly'
  'slurring every word'
  'in a drunken stupor'

Some supported grammatical forms:

<user> <says>, <mood>, <terminating-text>
<user> <says> <mood>, <terminating-text>
<non-terminating-text>, <user> <says> <mood>.
<user> <says>, <terminating-text>
<non-terminating-text>, <user> <says>.

Emotes

There will be hundreds of emotes supported for the player run species. Many, if not all, can be activated using a chat command. Common acronym forms (such as /LOL) will be supported where possible. See the chapter on Emotes for details.

Filtering

SWO will be using client-side filtering; technically the simplest method, allowing the server to pass on unedited messages to the receiving clients. Those chat clients produce either Raw, Chat, or Prose text while also filtering obscenities. Bandwidth usage varies with the size of the message and the receivers. To reduce bandwidth requirements, the server will compress the unedited text once and send that to one or more receivers.

Ignore List

Ignoring should ignore any player-initiated messages from the ignored player. Though I can't think of an example offhand, player-initiated text that would *not* be ignored would include something like spell words, i.e. anything that would put the ignoring player at a tactical disadvantage. The text from ignored players includes spatial speech, chat messages, instant messages, and probably mail. Ignoring any or all of these message types should be optional in the ignore list interface.

- The message comes in but is totally ignored and not buffered by the client side.
- Individuals or groups of individuals can be placed in a player's ignore list. Such as a clan and all the individuals belonging to the clan.
- Ignored individuals on screen may have an icon next to them.
- The following options should exist:
    - Ignore Speech
    - Ignore Shouts
    - Ignore Tells
    - Ignore Chat
    - Ignore in Channel (whatever current active channel is)
    - Ignore All
- Need standard windows interface for editing the ignore list. You should also be able to double-click or <select> people in the list to pull up additional information about that person, as well as be able to modify what rules you are currently ignoring them under.
- Ignore system should tie into friends list in that if you choose to ignore someone that is present in the friends list, a warning comes up for verification, and then pulls the ignored person from the friends list as well.
- If you message someone who has you on their ignore list, you get no feedback that they didn't receive your message. If you message someone that is temporarily ignoring you/has squelched you, you are told that a player has temporarily squelched you.
- On a global channel, you are not told that you are being ignored, but on a private channel you are. Messages sent to one client tell you if you are being ignored or not. Messages to multiple clients give no such ignore-status feedback. These settings also only apply on a per-character basis, not a per account basis. You should be able to import these settings from your character or another account.

Squelch

Squelch is a temporary form of ignore. It is not stored anywhere and if you log out and log back in, you lose your squelch settings. Note that this is **not** the same as GM squelching.

You can bring up a listing of everyone on the channel, the right-click or <select> people on the channel to squelch him or her. There are also SQUELCH ALL and UNSQUELCH ALL buttons. This interface for squelch would be primarily used on group channels (clan and party). In addition, it should have some squelch grouping that takes into account any command structure within that group (in a squadron, you'd have the squadron-leader, and several wing-leaders, then the wing-men assigned to the leaders. You'd be able to squelch out all the wingmen, or listen just to the squad leader, etc.)

- You can also squelch individuals.
- Squelch can be based on time or off of session.
- You can squelch combat messages. You can't squelch Server/Supervisor system messages.
- SQUELCH and IGNORE LIST have the same capabilities, but one is temporary but the other is not. They are the same thing, but depend on the setting of a 'temporary-session/temporary-time/permanent' option.

Special Features

The following special features are recommended for the primary release of the game in some form.

Command completion

Priority: Higher

The number of chat commands available to the veteran user, while powerful, is still quite staggering. We can easily provide a mechanism for command auto-completion to more quickly facilitate the entry of chat communications.

As the player types text at the chat prompt, the text entry subsystem should test the message being entered against all chat commands and possibly any previously entered messages.

The prompt could display matching text in bold (or dark color) and the recommended completion text in non-bold (or lighter color.) The cursor will remain at the correct position for the player to continue entering text.

Say Example:

/sa | y

In this example, the player has entered '/', 's', and 'a'. The cursor remains at the end of the entered text, and the recommended text is displayed in the prompt field after the cursor. If we use an underscore cursor instead of an insertion cursor, the cursor will be placed under the first character of the recommended text.

At this point the player may hit one of several auto-completion command keys. The following auto-completion features should be supported:

- Continue entering text with auto-completion still enabled. (default)
- Auto-complete and continue entering text.
- Auto-complete and send the text.
- Delete the recommended text and continue entering with auto-completion disabled for this message.

WYSIWYG text pre-filtering

Priority: Lower

Like auto-completion, this would pre-filter text before sending it, showing you what players will read. This may or may not work in conjunction with the spell checking system. Optionally, players could choose which style they wish to see in pre-filtering mode: Raw, Chat, or Prose.

Spell Checking

Priority: Moderate

As this feature will be needed for in-game mail, it makes sense to choose a package capable of prompt style spell checking as well.

Logs / Log appending vs. Log instancing

A log or journal option should be available for each channel the player joins. In addition, an option should exist to indicate whether the log is appended or overwritten each time the player joins the channel. Log files should be maintained on the client system, exclusive of the more secure logging system used to lodge abuse complaints.

System Messages

Priority: Required

System messages should be displayed in the Active Channel, or perhaps all channels. If the chat interface is not enabled, system messages should be displayed prominently in the current game view.

Macro Messages

Priority: Higher

Macro messages are pre-canned audio/text messages and possibly emote content in the ground game. The most common use for these in earlier games has been taunts and buddy commands used in first-person shooters, flight games with wingmen, and squad-based games.

Fictional Language Translation

Priority: Higher

Many of the player-run species in Star Wars have their own language other than 'Basic', including Wookiee and Rodian. We can support a language system with simple, word-based text mangling that 'translates' words based on the skill of the receiver. For security reasons, the language mangling must occur on the server. This same un-mangling system works with a translator item, should such a gizmo exist in the game universe.

Minimally, the language system should provide translators to encompass the following player-run species:

- Bothan
- Human
- Mon Calamari
- Rodian
- Trandoshan
- Twi'lek
- Wookiee
- Zabrak

Future updates to the live system should include language translators for the various NPC species as well.

Conversational Audio Queueing ("Sims"-speak)

Priority: Lower

For spatial speech, and in conjunction with moods/emotes/gestures, overlay audio snippets of a player speaking in a tone appropriate for the message. At the simplest level, these could be controlled by players like gestures and emotes. Audio snippets would be mixed in rooms with lots of people speaking. Each species would have its own set of conversational snippets.

GM Squelch Tools

Priority: Required

Provide tools for GMs to squelch players from communicating to a) other players, b) in chat channels, and c) event support staff.

Squelched players have an icon or text above their head indicating they are squelched.

Squelching can be permanent or time-based and therefore temporary.

Spatial Spam Control

Priority: Required

In spatial speech, the primary goal of spam control is to limit the number of messages over a period of time. This limit alone is not enough, if a grief player chooses to spam a full buffer of text every few seconds. A few lines of garbage text, repeated over time, can wipe out a real conversation. And while allowing players to ignore these other players offers a great tool, we can also provide the mechanics to rebuke and dissuade players from acting in this manner.

The first line of defense is limiting the text entry character buffer length to a size that allows players to be expressive and informational, without being able to write more than a large paragraph. With the right statistics gathering during beta, we may be

able to set this number without having to guess at a good size for release.

The second line of defense is limiting the number of messages over time, a function controlled on both the client and server, to avoid client hacking. Once again, good statistics gathering will help us determine a number.

We should also limit the number of characters sent over time with high and low 'water' marks. The value of the high water mark should be lower than the size of the text entry buffer. If the player sends more characters in the high water mark, then an internally held value must reach the low water mark before the server processes other messages. In essence, the server will hold back the text until the low water mark is reached.

When players abuse the high-water mark, and until the internal variable reaches the low water mark, subsequent messages from the player will be met with responses such as,

```
Your voice has become too hoarse to continue. (Text queued for # seconds.)
```

The server should check for repeat messages and squelch abusing players for a short period.

Finally, as the last line of defense, any account abusing the system with denial-of-service types of attacks such as a hacked client spewing chat packets, should be noted to a GM and the client kick/banned until a GM can review the account and incident.

Use cases should include:
1. Multiple, large and differing messages sent of a short period.
2. Multiple, duplicate messages over a short to moderate period.
   ($#&@ YOU!↵ $#&@ YOU!↵ $#&@ YOU!↵…)
3. Multiple short messages over a very short period of time.
   (a↵b↵c↵1↵2↵3↵4↵…)

Channel Spam Control

Priority: Required
Similar to spatial speech, spam control should be handled in channels as well.

Moderated Spam Control

Priority: Moderate (no pun intended)
Allow channel moderators so set spam control values, including high and low water marks. Perhaps set actions to take (kick/ban) from channel when abuse occurs. This would not work in public channels, only party, guild, clan, and other moderated channels.

Interface

The purpose of this document is to provide an overview of chat systems and their requirements without getting into too much interface detail. Future documents (Chat Systems and/or Interface) will include full details on the look and feel and functionality required of the graphical and command interfaces.

General Notes

Throughout all chat systems, for every command-line action that can be used, there should be an equivalent interface option.

Older commands should be scrollable at command-line prompts, or perhaps selected using a list box, and fully editable before sending.

Text Entry Notes

Toggle-able auto-word finishing.

Two limits of how much text can be typed in on one line.... One is that no individual message can be longer than 512 characters. The other limit is a characters per second limit. This limit can be overrun in the message that you are currently typing, but whatever additional characters you type are pulled out of the next messages per-second limit...perhaps around 100characters per second.... If you input 500 characters, you can't send/transmit any additional text for another 4 seconds. You could enter it however... Home, End, Word left, Word right (CTRL-right), Cut, Copy, 'n Paste. Undo...Insert/Over-write, spell-checking... on the fly misspell high-lighting...

Cut, undo, and redo function on the active entry line.

Text Buffering Notes

- Buffered for X number of lines
- Player determined
- Every message displays the name of the sender and might also display a time-stamp (optionally defaulting to off, and handled on the client side.)
- Scroll up and down through the lines that have been typed thus far using either keyboard or mouse interface, possibly joystick or joystick hat also, and make it remappable.
- Currently selected line should be highlighted somehow from the other lines.
- Add a scroll bar ... all standard windows interface things.... Scroll bar, etc.
- When you scroll halfway up through the list
- As soon as you scroll up through the listing any new lines added does not force the buffer display down to the bottom to look at them, but rather simply add them to the bottom of the buffer.
- If you scroll up to look at a message stored in the buffer, the text display should optionally split into two windows, with one showing your current place in the buffer, and the bottom window showing the new text that is currently being added.... The current window would subdivide.
- There should be several ways to send a message to someone who already sent one to you... it could be right-clicking on a name in the message buffer.... Or it could be set to left click with right-click bringing up options.... There needs to
- Optionally, you would send back to them the same way they sent to you, or using some other method... would depend on how the options for the buffer were set up, or upon how the interface was set up... (left click to simply send back through same method, right click to /chat or a method of your choosing)
- Buffer has the filter active.

Text Filtering Notes

- Should include all industry standard words and word-detection-algorithms....
- Word replacement... replaces words with either symbols or words...
- Filter definitely filters what is received before it is displayed for you....
- Optionally this could filter what you type as you type it, when you space to type another word. You get a bit of a WSYWIG effect.
- As an option, the player can add words to filter out, as well as add words to replace the filtered words.
- Perhaps have the server be able to poll people's dictionaries to see what people are adding to them. This information could be used to add for patching...
- Dictionary size might be unlimited. Polling of dictionaries could take place at logon; the server might skip a file if it is beyond a certain size

Combat Notes

- You could get combat affects information on the conditions of your allies on their channel that you are all on, but this level of info can be filtered out.

- Categorize messages by general category and then prioritize them. In combat, anything that is related to you is highest priority. You indicate priority and category by color and font and font-size, and bold/highlight.

Assets Required

Technical Notes:

The server will be required to pass on the exact text sent by a player so that receiving clients may produce either prose or raw text based on the receiver's option settings.

The generated prose is written to both the chat window and the journal file (log) for the character.

Mood and Set Method settings are saved between gaming sessions, and must be saved per character, not per account.

Whitespace handling between command and text in raw mode is still TBD.

Command System Notes

Listed below is a subset of the final list of chat system commands.

Chat Channels

Peer-to-Peer

```
/CHAT target [msg]
/INVITE target [msg]
```

Channel Commands

```
/JOIN [channelname]
/JOIN CLAN
/JOIN GUILD
/JOIN PARTY
/AFK [message]
/NICK [name]
/WHISPERTO target[,target,...] msg
```

Channel Moderator Commands

```
/KICK [target]
/BAN [target]
/MODERATOR [target]
/MODERATE [on|off]
/VOICE [target]
/LIMIT [count]
/TOPIC [topic]
/PASSWORD [password]
```

Spatial Chat

```
/WHISPER
/YELL
```

Moods

```
/MOOD                       ; reports current mood
/MOOD moodname              ; set mood
/SAYMODE                    ; reports current say method
/SAYMODE saymode            ; set say method
/HELP MOODS                 ; list of moods
/HELP SAYMODES              ; list of say methods
```

Instant Messaging

```
/TELL target[,target,…] msg
```

Ignore List

```
/IGNORE target [options]
/UNIGNORE target [options]
/SQUELCH target
```

Revision Log

1.	01-Aug-00	Initial revision.
2.	18-Aug-00	Minor changes to Chat Channels, Text Generation Options, Say examples, Say Alternate examples, and Sticky Mood commands.

EMOTE

The following list contains some of the emote animations that should be supported. Emotes in CAPS could be considered an emote category.

CLAP	Applaud									
BACKRUB	Massage									
SMILE	Beam	Grin	Smirk							
HUG	Bearhug	Embrace	Cuddle	Squeeze	Tackle					
BITE	Nibble	Nip								
WIELD	Brandish									
BECKON	Summon									
COWER	Cringe	Grovel?								
CACKLE										
LAUGH	Chortle	Chuckle	Giggle	LOL	Snicker					
DISAGREE	Shake									
DUCK	Bob									
TOUCH	Feel	Fiddle?								
FROWN	Glare									
GROAN	Moan									
GROPE	Fondle	Pinch	Tickle	Twiddle						
HAIL	Hello	Wave								
HOP	Jump									
POSE	Model									
NOD	Agree									
PAT	Comfort	Reassure	Tap							
POUT	Mope	Sulk								
PONDER	Puzzle	Think	Question							
PUSH	Poke	Press	Prod	Tag						
SHIVER	Quiver	Shudder	Squirm	Startle	Tremble	Twitch	Convulse	Wiggle	Wobble	
KISS	French?	Smooch								
SMELL	Sniff	Sniffle								
GROWL	Snarl									
NUZZLE	Snuggle									
PET	Stroke									
THREATEN	Menace	Warn								
PULL	Tug									
SLAP	Whap									
WHINE	Whimper									
STARE	Gaze	Zone								
THANK	Handshake?									
Bat	Blink	Blush	Bonk	Bounce	Bow	Comb	Cough	Count	Cover	Burp
Cheer	Cry	Curtsey	Dance	Defend	Drool	Deride	Eyeroll	Eyebrow	Face	Flip
Fiddle	Fidget	Flex	Flourish	French	Gasp	Gloat	Grumble	Gulp	Hang	Hear
Hiccup	High5	Hiss	Handkiss	Handshake	Ignore	Kneel	Leap	Lick	Mourn	Nudge
Pant	Peer	Point	Polish	Ponder	Puke	Punch	Quote	Raise	Ruffle	Sacrifice
Scream	Scratch	Sharpen	Shrug	Sigh	Slurp	Snap	Sneeze	Snore	Snort	Spank
Stagger	Step	Stir	Stomp	Strangle	Stretch	Strut	Swoon	Tongue	Turn	Wait
Summon	Dismiss	Whistle	Wince	Wink	Worship	Wrinkle	Yawn	Tip		
Caress	Faint	Twirl	Page	Salute	Spit	Waggle	Fume			

While this list looks daunting from the perspective facial and body animation, many of these use the same animation regardless of the emote target. Some require different animation depending on the target. Still others are blends of two emotes and a pose (mood).

The engine and tools must provide a way for artists and designers to define emotes through blending animations. In addition, there must be a way to specify the target of an animation through a hardpoint or bone to animate handshakes, kisses, petting, patting, et. al.

Minimally, the engine should support blending a combination of four to six skeletal and facial animations.

MOODS

The Moods system provides the ability to present moods within the context of the spatial chat and prose generation for the interactive chat system and journal.

Vision

We will support player-set moods and other such indicators of speech. Players will be able to tell whether the player is trying to sound angry, or sad, or joking. We will also visually reflect this in the player's avatar via the idle animations, which will match the mood selected. Mood selection will be done with a simple graphical interface for basic moods, and with command line prefix commands (e.g., *angry say*) for power users.

In addition, we will parse out emoticons and use them to affect the text output and the avatar's animations. Putting in an unhappy face in your speech, such as ":(" will result in your avatar's shoulders slumping.

Lastly, text output will be optionally configurable to read like prose, with speech tags appearing before, after, and in the middle of text, some extent of paragraphing, and other cues that make the output easier to read for those accustomed to narrative fiction. Players can always resort to standard mud-style mud output if they choose.

The full array of speech system commands will be supported, along with alternates such as mumble, growl, grumble, etc, to allow greater expressiveness.

- There must be a wide range of in-game communication systems.
- Players need tools to shape their own experiences.
- Amy Jo Kim's principles for community building:
  - We will encourage appropriate etiquette.
- Humor and personality will be important in the game design.
- We must have full support for localization, including Unicode support on client and server.

System Details

The default mood is neutral and provides no prose context. Moods can be set/entered via an interface or chat command.

Mood Commands

Support dozens of moods through the use of multiple chat commands, each with one or more grammatical variations.

Example (assumes prose mode):

A human types:

```
/gruff say Don't insult us. You just watch yourself. We're wanted men. I have the death sentence in twelve systems.
```

Everyone reads:

"Don't insult us," a human says gruffly. "You just watch yourself. We're wanted men. I have the death sentence in twelve systems."

Luke types:
`/contrite ' I'll be careful then.`

Everyone reads:
"I'll be careful then.", Luke says contritely.

A human types:
`/angry say You'll be dead!`

Everyone reads:
Angrily, a human says, "You'll be dead!"

"Sticky" moods

Some players may want their characters to walk around happily or gruffly in all gaming sessions, or their character may wish to present a bad mood.

To see your current mood, type:
`/mood`

To see a list of moods, type:
`/help moods`

To return to a neutral mood, type:
`/mood -`

To set the default mood for a character, simply type:
`/mood` *`moodname`*

Examples of some of the many supported moods and prose follow:

- angry
  'angrily'
  'furiously'
  'with a cold, angry glare'

- bored
  'bored'
  'uninterested'
  'bored stiff'

- drunk
  'drunkenly'
  'slurring every word'
  'in a drunken stupor'

Some supported grammatical forms:

<user> <says>, <mood>, <terminating-text>
<user> <says> <mood>, <terminating-text>

<non-terminating-text>, <user> <says> <mood>.
<user> <says>, <terminating-text>
<non-terminating-text>, <user> <says>.

Core Requirements

The following mood forms should be supported by the prose generation system, and backed by character animation where appropriate.

abashed	absentminded	aghast	agonized	agreement
amazed	amused	angry	angstful	annoyed
approving	ashamed	bewildered	bitter	bloodthirsty
bored	brave	callous	careful	careless
casual	childish	clinical	cocky	cold
compassionate	condescending	confident	confused	content
courtly	coy	crude	cruel	curious
cynical	defensive	depressed	devious	dimwitted
disagreement	disappointed	discreet	disgruntled	disgusted
dismayed	distracted	doubtful	dramatic	dreamy
drunk	earnest	ecstatic	embarrassed	emphatic
encouraging	enthusiastic	evil	exasperated	exuberant
fanatical	foolish	forgiving	frustrated	guilty
greedy	happy	harsh	helpful	honest
hopeful	hopeless	humble	hungry	hurt
hysterical	indifferent	impatient	imploring	impressed
incredulous	indignant	innocent	insane	interested
intrigued	jealous	joyful	kind	knowing
lazy	lofty	lonely	loud	loving
lustful	mean	meek	melancholy	mischievous
musical	mysterious	mystified	nervous	neutral
offended	optimistic	pained	patient	pedantic
pensive	pessimistic	petulant	philosophical	pitiful
pitying	playful	polite	pompous	pouting
precocious	proud	provocative	protective	puzzled
quick	quiet	rebellious	regretful	relieved
reluctant	resigned	respectful	restless	romantic
rude	sad	sarcastic	scared	scolding
scornful	secretive	seductive	serious	shameless
shocked	shy	sincere	sleepy	slow
sly	smug	snobby	soft	sorry
spiteful	stupid	stubborn	subject	subtle
sullen	surprised	suspicious	sweet	taunting
teasing	terrified	thankful	thoughtful	timid
tired	tolerant	uncertain	understanding	unhappy
unwilling	victorious	warm	whiny	wicked
wise	wistful	worried		

Revision Log

1. 17-Aug-00 Initial revision.

SAY COMMAND

Say commands communicate text through the Avatar using spatial speech, regardless of the active channel. Therefore, at present, this design limits say commands to spatial speech only. Our goal is to provide say commands, moods, and emotes, in channels and spatial speech simultaneously – without having to change to a spatial speech channel. The prose generated by spatial speech may be output to a journal.

Vision

We will support player-set moods and other such indicators of speech. Players will be able to tell whether the player is trying to sound angry, or sad, or joking. We will also visually reflect this in the player's avatar via the idle animations, which will match the mood selected. Mood selection will be done with a simple graphical interface for basic moods, and with command line prefix commands (eg, *angry say*) for power users.

In addition, we will parse out emoticons and use them to affect the text output and the avatar's animations. Putting in an unhappy face in your speech, such as ":(" will result in your avatar's shoulders slumping.

Lastly, text output will be optionally configurable to read like prose, with speech tags appearing before, after, and in the middle of text, some extent of paragraphing, and other cues that make the output easier to read for those accustomed to narrative fiction. Players can always resort to standard mud-style mud output if they choose.

The full array of speech system commands will be supported, along with alternates such as mumble, growl, grumble, etc, to allow greater expressiveness.

- There must be a wide range of in-game communication systems.
- Players need tools to shape their own experiences.
- Amy Jo Kim's principles for community building:
  - We will encourage appropriate etiquette.
- Humor and personality will be important in the game design.
- We must have full support for localization, including Unicode support on client and server.

System Details

Core Requirements

Say Commands

By default, the "/say" command uses the current "sticky" settings for both Say Methods and Moods. The default Say Method is "says" and the default mood is neutral, meaning no mood is implied in the generated prose. Moods and Say Methods is described in the following sections.

An alternative to the say command is the single quote ' at the *beginning* of the chat line or chat line parameter (see Moods.) White space between the ' and subsequent text may be deleted by the prose generator.

Luke types:
```
/say Hello
```

Everyone reads:
```
"Hello," Luke says.
```
Alternatively, everyone reads:
```
Luke says, "Hello."
```

Alternatively, those with raw more enabled read:
```
Luke: Hello
```

Luke types (note the optional say command):
```
'Hello. My name is Luke
```

In prose mode, can produce many variations:
```
"Hello. My name is Luke," says Luke.
"Hello," says Luke. "My name is Luke."
Luke says, "Hello. My name is Luke."
```

In raw mode (option), saying something in chat will reproduce the exact text typed by the sender on the receiver's client.

Luke types:
```
/say Hello
```

Clients in raw mode see:
```
Luke: Hello
```

Say Alternates

In addition to /say and the ' (single quote) versions of the say command, other forms of say should include:

say	add	admit	apologise	apologize
assert	babble	cackle	carol	chatter
complain	coo	declare	demand	describe
drawl	drone	enthuse	exclaim	explain
groan	growl	grumble	grunt	howl
indicate	moan	mumble	mutter	note
plead	prattle	protest	quote	ramble
request	retort	scream	smirk	snarl
sneer	state	stutter	threaten	utter
wail	whimper	whine		

Luke types:
```
/whine But I was going into Toshi Station to pick up some power converters...
```

Everyone reads:

"But I was going into Toshi Station to pick up some power converters," Luke whines.

Alternatively, everyone reads:

Luke whines, "But I was going into Toshi Station to pick up some power converters…"

Alternatively, those with raw more enabled read:

Luke: But I was going into Toshi Station to pick up some power converters…

In prose mode, can produce other variations:

"But I was going into Toshi Station to pick up some power converters," whines Luke.

"But I was going into Toshi Station to pick up some power converters," Luke whines.

Luke whines, "But I was going into Toshi Station to pick up some power converters…"

Note: Yelling may be primarily a spatial speech operation, however it might also be convenient to support yelling as a say alternate in other chat channels as well, providing context in prose mode and journals.

"Sticky" Say Commands

Like moods (below), some say alternates can be made 'sticky' with the /saymode command. Their character will walk around grumbling or cooing from session to session.

Technical Notes:
The server will be required to pass on the exact text sent by a player so that receiving clients may produce either prose or raw text based on the receiver's option settings.
The generated prose is written to both the chat window and the journal file (log) for the character.
Mood and Set Method settings are saved between gaming sessions, and must be saved per character, not per account.
Whitespace handling between command and text in raw mode is still TBD.

Revision Log

2. 17-Aug-00 Initial revision.
3. 18-Aug-00 Minor changes to Say and Say Alternate examples